OMD Documentation

Release 1.0

OMD

December 17, 2014

Contents

1 User Guide **3**
1.1 Parts Implemented by Kerem Adalı . . . 3
1.2 Parts Implemented by Arda Özdere . . . 7
1.3 Parts Implemented by Tamer Temizer . . . 12
1.4 Parts Implemented by Haydar Şahin . . . 15

2 Developer Guide **27**
2.1 Parts Implemented by Kerem Adalı . . . 28
2.2 Parts Implemented by Arda Özdere . . . 31
2.3 Parts Implemented by Tamer Temizer . . . 37
2.4 Parts Implemented by Haydar Şahin . . . 39

Team OMD

Members

- Kerem Adalı
- Arda Özdere
- Tamer Temizer
- Haydar Şahin

OMD - Online Museum Database

Online Museum Database is a website for finding pretty much any information about museums. Information can be found about Museums, Artworks, Artists, Museum Managers, Artwork Collectors and even important Visitors of Museums.

Contents:

CHAPTER 1

User Guide

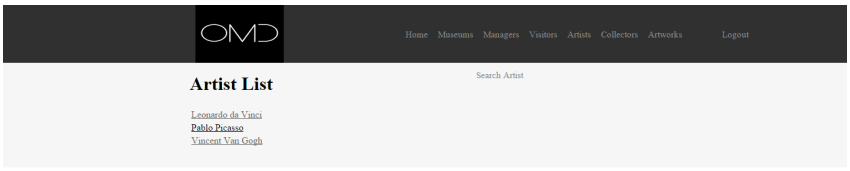
OMD - Online Museum Database

Online Museum Database is a website for finding pretty much any information about museums. Information can be found about Museums, Artworks, Artists, Museum Managers, Artwork Collectors and even important Visitors of Museums.

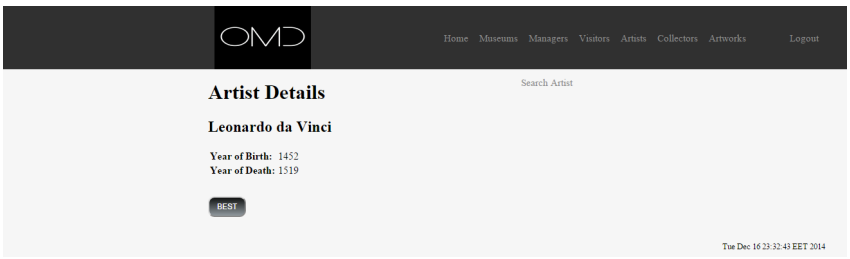
1.1 Parts Implemented by Kerem Adalı

1.1.1 Artist (User View)

After clicking the "Artists" link at the top, site directs you to the page where you can find the artist database of the OMD. General structure of the page is same with the remaning of the pages.



Names of the artists will lead you to the basic info page of the related artist. Here you can see the birth and death years of the artist. The best button at the bottom will be explained at the Users Interaction with Artist part.



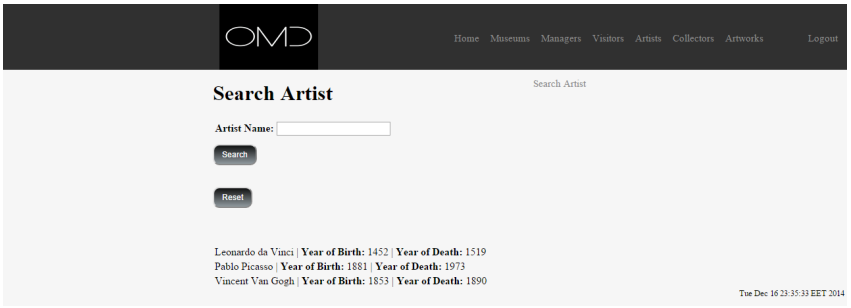
Search

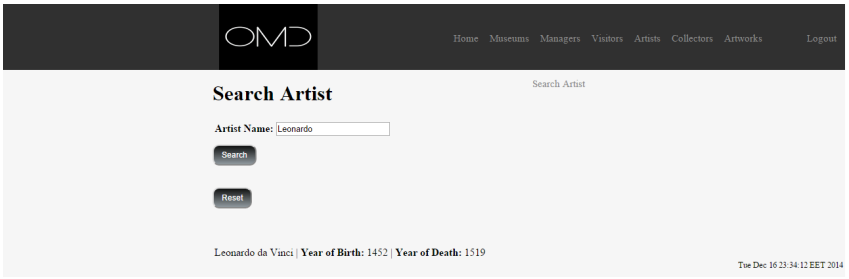
Search Page brings the info of the desired artists if they exists.

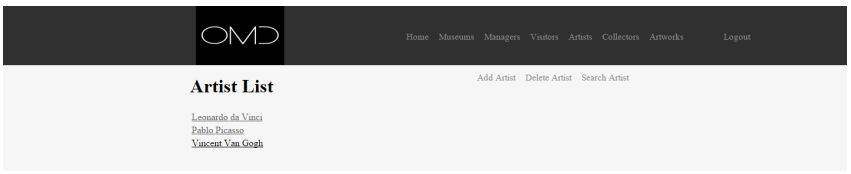
Reset button resets the page to its original form.

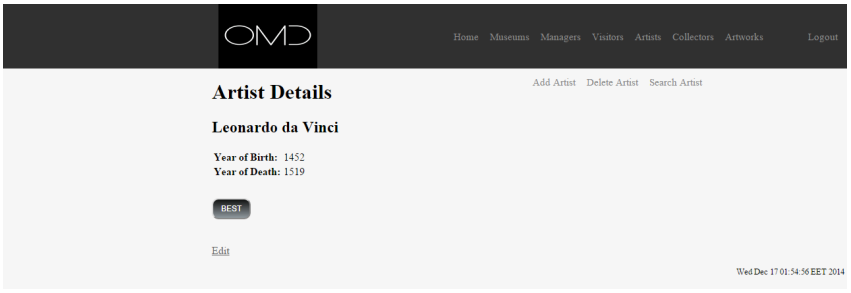
1.1.2 Artist (Editor View)

General structure of this page is same with the User View except the Add, Delete and Edit options.


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Artist
Search Artist
Artist Name:
Search
Reset
Leonardo da Vinci | Year of Birth: 1452 | Year of Death: 1519
Pablo Picasso | Year of Birth: 1881 | Year of Death: 1973
Vincent Van Gogh | Year of Birth: 1853 | Year of Death: 1890
Tue Dec 16 23:35:33 EET 2014

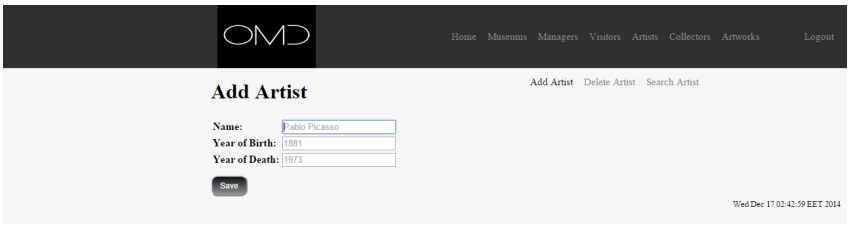

OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Artist
Search Artist
Artist Name: Leonardo
Search
Reset
Leonardo da Vinci | Year of Birth: 1452 | Year of Death: 1519
Tue Dec 16 23:34:12 EET 2014


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Artist List
Add Artist Delete Artist Search Artist
Leonardo da Vinci
Pablo Picasso
Vincent Van Gogh


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Artist Details
Add Artist Delete Artist Search Artist
Leonardo da Vinci
Year of Birth: 1452
Year of Death: 1519
BEST
Edit
Wed Dec 17 01:54:56 EET 2014

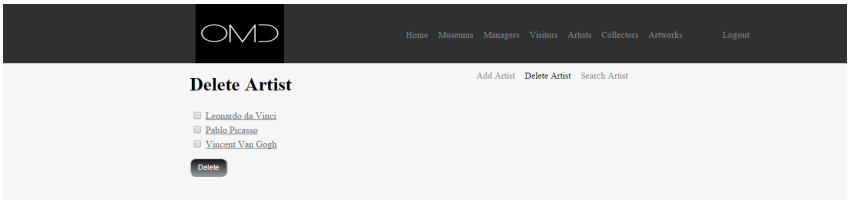
Add

You can reach this page from the “Add Artist” link and on this page you can simply add new artists just by filling the empty boxes and clicking the save button.



Delete

When the “Delete Artist” link clicked OMD directs you to this page and here you can delete any artist by checking the boxes and clicking delete. Multiple deletion is also possible.



Edit

You can open this page by clicking the edit link at the bottom of the Display Page of the artist you want to edit and change the information just by rewriting it and clicking save button.

OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Add Artist Delete Artist Search Artist
Add Artist
Name: Leonardo da Vinci
Year of Birth: 1452
Year of Death: 1519
Save
Wed Dec 17 02:42:22 EET 2014

1.1.3 Collector (User View)

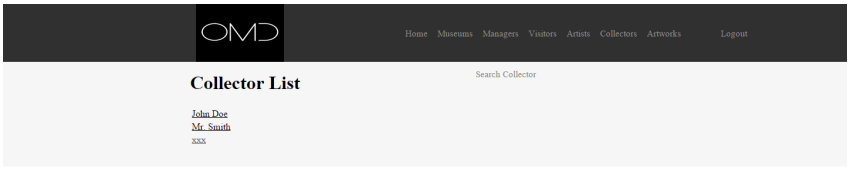
After clicking the “Collectors” link at the top, site redirects you to the page where you can find the collector database of the OMD. General structure of the page is same with the remaning of the pages.

Names of the collectors will lead you to the basic info page of the related collector. Here you can see the favorite artist of the chosen collector. Name of the artist will lead you to its info page.

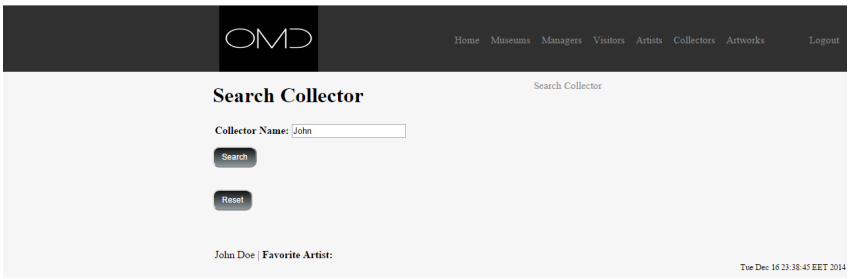
Search

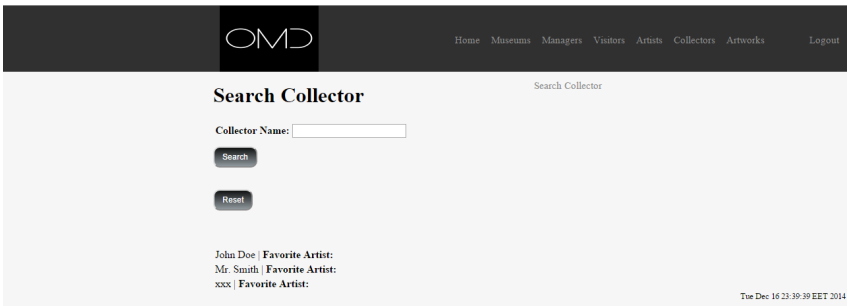
Search Page brings the info of the desired collectors if they exists.

Reset button resets the page to its original form.


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Collector
Collector List
John Doe
Mr. Smith
xxx

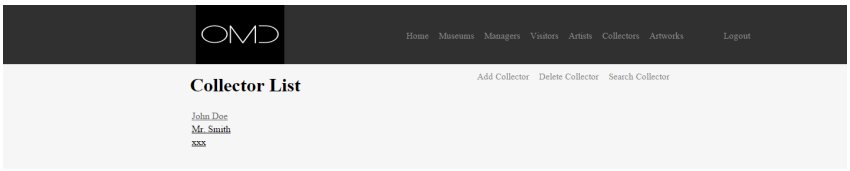
OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Collector
Collector Details
John Doe
Favorite Artist: Leonardo da Vinci
Tue Dec 16 23:37:36 EET 2014

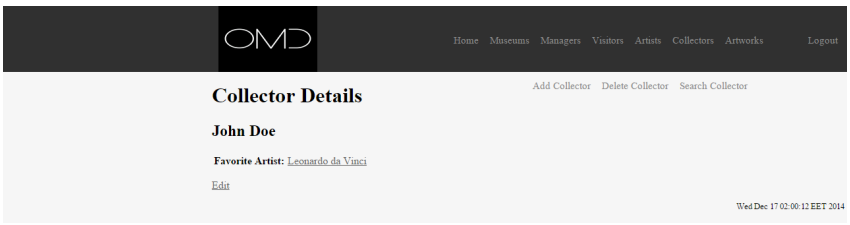

OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Collector
Search Collector
Collector Name: John
Search
Reset
John Doe | Favorite Artist:
Tue Dec 16 23:38:45 EET 2014


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Collector
Search Collector
Collector Name:
Search
Reset
John Doe | Favorite Artist:
Mr. Smith | Favorite Artist:
xxx | Favorite Artist:
Tue Dec 16 23:39:39 EET 2014

1.1.4 Collector (Editor View)

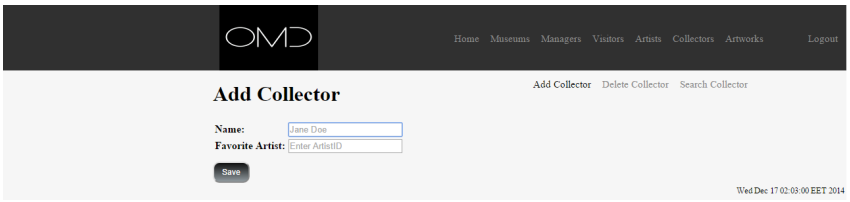
General structure of this page is same with the User View except the Add, Delete and Edit options.





Add

You can reach this page from the "Add Collector" link and on this page you can simply add new collectors just by filling the empty boxes and clicking the save button.



Delete

When the "Delete Collector" link clicked OMD directs you to this page and here you can delete any collector by checking the boxes and clicking delete. Multiple deletion is also possible.

Edit

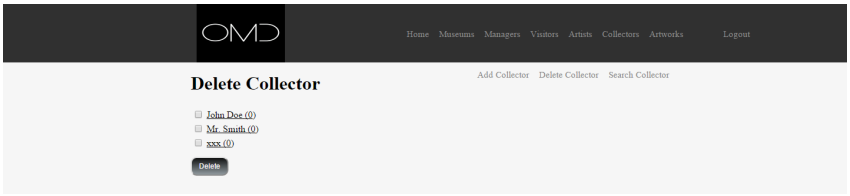
You can open this page by clicking the edit link at the bottom of the Display Page of the collector you want to edit and change the information just by rewriting it and clicking save button.

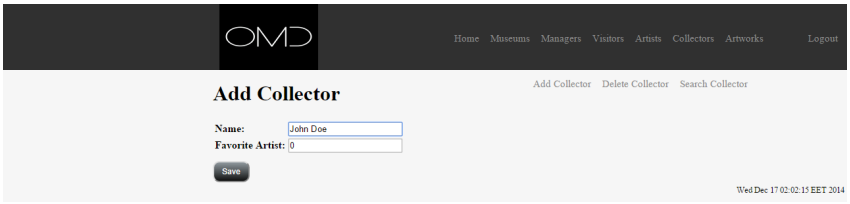
1.2 Parts Implemented by Arda Özdere

1.2.1 Museums

When the 'Museums' link is clicked on the panel, the user gets the page that displays the list of the museums. If you are an editor, on the top right of this page there are links to the options for listing, adding, deleting or searching the museums. If you are a basic user, you can only see the search option. The 'Museum List' link is directed to the current page.

The museum list on the page consists of links that directs the user to the display page of the selected museum.





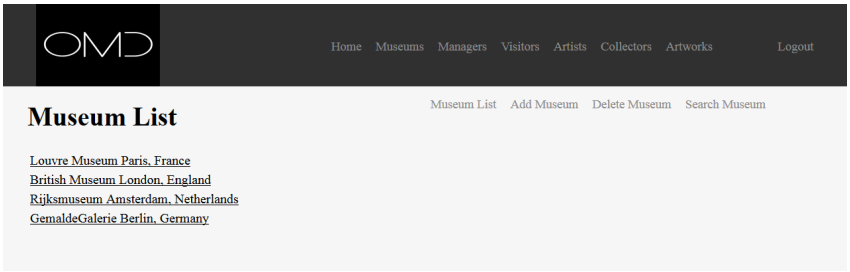

OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Museum List Add Museum Delete Museum Search Museum
Museum List
Louvre Museum Paris, France
British Museum London, England
Rijksmuseum Amsterdam, Netherlands
GemaldeGalerie Berlin, Germany

Figure 1.1: Figure 1: Museum List Page

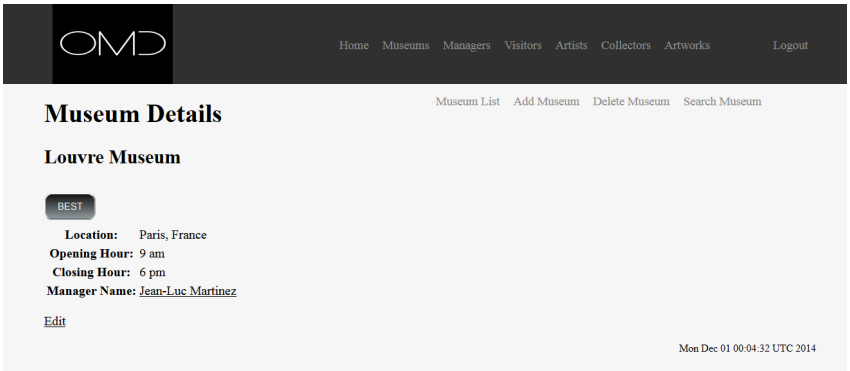


Figure 1.2: Figure 2: Museum Display Page

In the display page, the user can see the information of a museum which are the name, location, opening hour, closing hour and the manager of the museums. The 'BEST' button can be clicked to make it the users favourite museum, the name of the manager can be clicked to go to that managers display page or the 'Edit' button can be clicked to go to the edit page of the museum. The 'Edit' button is only visible to the editors, basic users are not able to edit data.

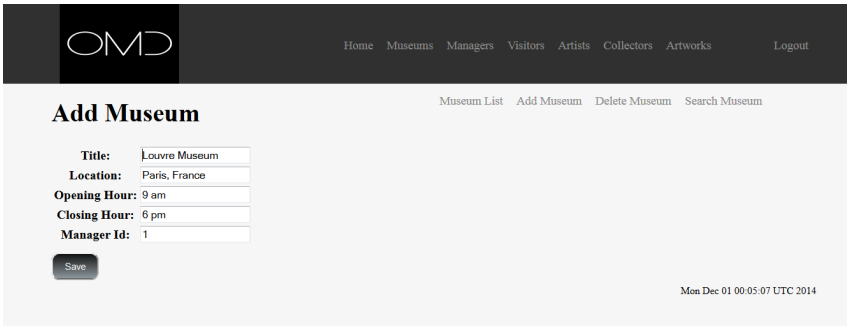


Figure 1.3: Figure 3: Museum Edit Page

If you are an editor, edit page makes it possible to edit the data of an existing museum by re-filling the textboxes and clicking save.

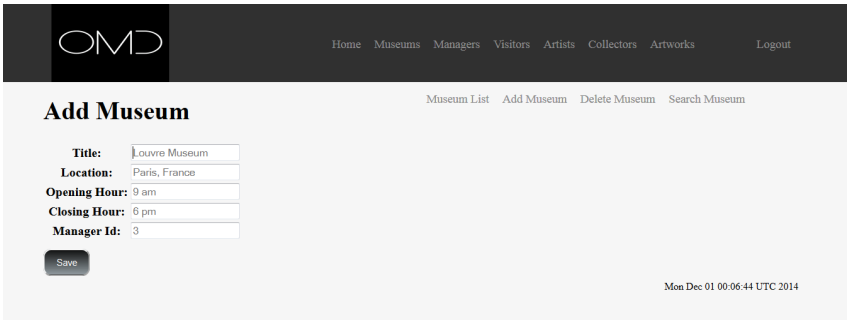


Figure 1.4: Figure 4: Museum Add Page

When the 'Add Museum' link is clicked the user gets the add page, where a new museum can be added by editors after filling the textboxes and clicking 'Save'.

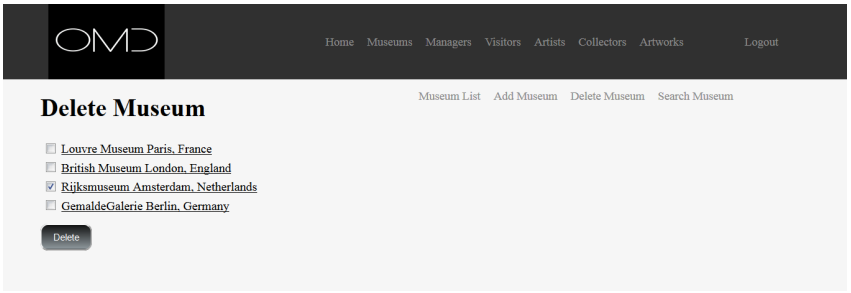


Figure 1.5: Figure 5: Museum Delete Page

Also if you are an editor, when the 'Delete Museum' link is clicked, the user is directed to the delete page consisting of a list of the museums with checkboxes near them and a 'Delete' button below. Selecting the desired checkboxes and clicking delete makes the user able to delete the selected museums.

Whether you are an editor or a basic user, if you want to search a museum, 'Search Museum' link can be clicked. After filling the textbox with the name that's wanted to be found, 'Search' makes it possible for the user to find out if there exists a museum with the given name in the database.

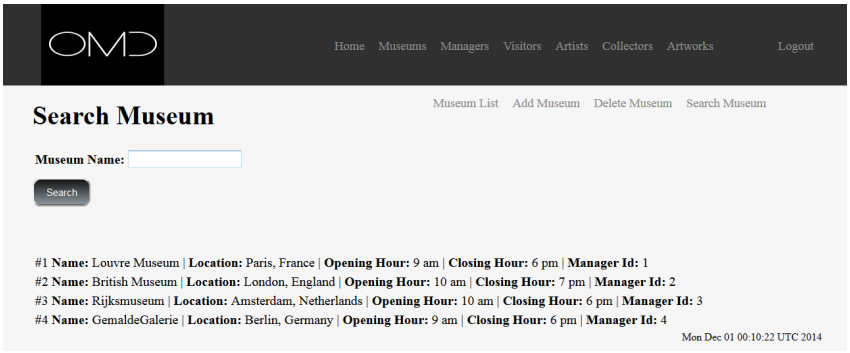

OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Museum List Add Museum Delete Museum Search Museum
Search Museum
Museum Name:
Search
#1 Name: Louvre Museum | Location: Paris, France | Opening Hour: 9 am | Closing Hour: 6 pm | Manager Id: 1
#2 Name: British Museum | Location: London, England | Opening Hour: 10 am | Closing Hour: 7 pm | Manager Id: 2
#3 Name: Rijksmuseum | Location: Amsterdam, Netherlands | Opening Hour: 10 am | Closing Hour: 6 pm | Manager Id: 3
#4 Name: GemaldeGalerie | Location: Berlin, Germany | Opening Hour: 9 am | Closing Hour: 6 pm | Manager Id: 4
Mon Dec 01 00:10:22 UTC 2014

Figure 1.6: Figure 6: Museum Search Page

1.2.2 Managers

When the 'Managers' link is clicked on the panel, the user gets the page that displays the list of the managers. If you are an editor, on the top right of this page there are links to the options for listing, adding, deleting or searching the managers. If you are a basic user, you can only see the search option. The 'Manager List' link is directed to the current page.

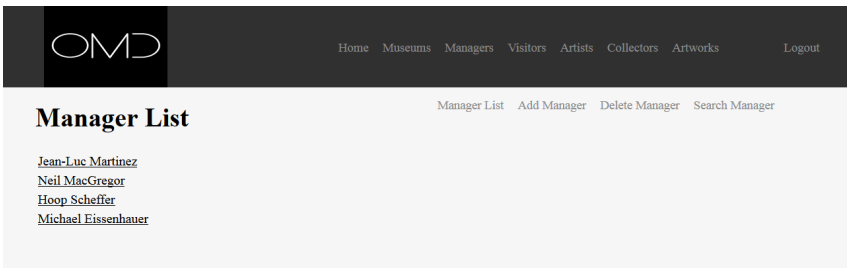


Figure 1.7: Figure 7: Manager List Page

The manager list on the page consists of links that directs the user to the display page of the selected manager.

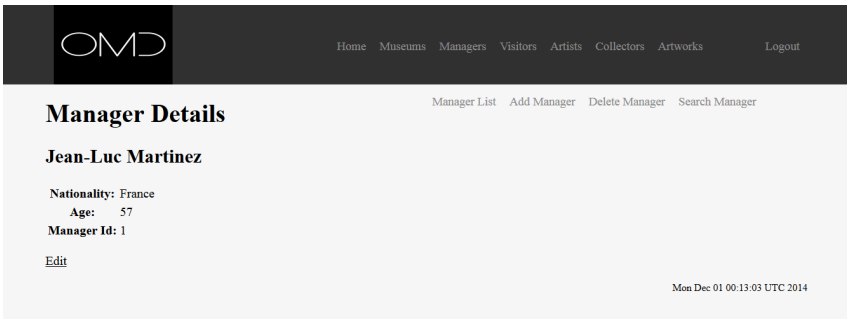


Figure 1.8: Figure 8: Manager Display Page

In the display page, the user can see the information of a manager which are the name, age, nationality and id of the manager. Id is displayed to make it easier for the user to find the managers while adding a museum as museum data includes a manager id. The 'Edit' button can be clicked to go to the edit page of the manager. The 'Edit' button is only visible to the editors, basic users are not able to edit data.

If you are an editor, edit page makes it possible to edit the data of an existing manager by re-filling the textboxes and clicking save.

When the 'Add Manager' link is clicked the user gets the add page, where a new manager can be added by editors after filling the textboxes and clicking 'Save'.

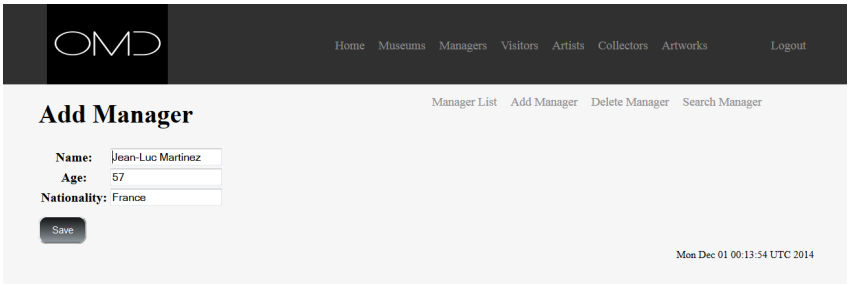


Figure 1.9: Figure 9: Manager Edit Page

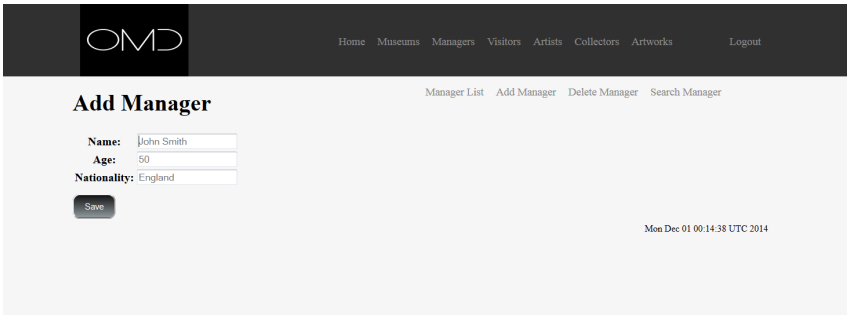


Figure 1.10: Figure 10: Manager Add Page

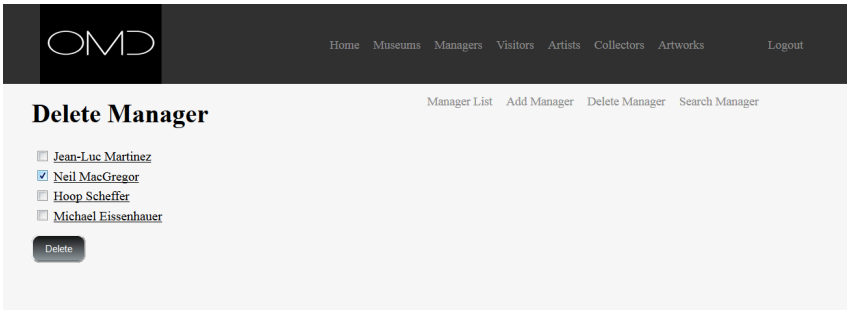


Figure 1.11: Figure 11: Manager Delete Page

Also if you are an editor, when the 'Delete Manager' link is clicked, the user is directed to the delete page consisting of a list of the managers with checkboxes near them and a 'Delete' button below. Selecting the desired checkboxes and clicking delete makes the user able to delete the selected managers.

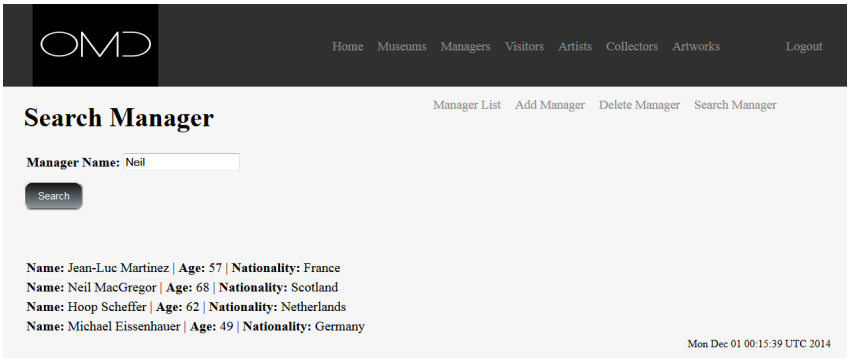

OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Manager List Add Manager Delete Manager Search Manager
Search Manager
Manager Name: Neil
Search
Name: Jean-Luc Martinez | Age: 57 | Nationality: France
Name: Neil MacGregor | Age: 68 | Nationality: Scotland
Name: Hoop Scheffer | Age: 62 | Nationality: Netherlands
Name: Michael Eissenhauer | Age: 49 | Nationality: Germany
Mon Dec 01 00:15:39 UTC 2014

Figure 1.12: Figure 12: Manager Search Page

Whether you are an editor or a basic user, if you want to search a manager, 'Search Manager' link can be clicked. After filling the textbox with the name that's wanted to be found, 'Search' makes it possible for the user to find out if there exists a manager with the given name in the database.

1.2.3 Visitors

When the 'Visitors' link is clicked on the panel, the user gets the page that displays the list of the visitors. If you are an editor, on the top right of this page there are links to the options for listing, adding, deleting or searching the visitors. If you are a basic user, you can only see the search option. The 'Visitor List' link is directed to the current page.

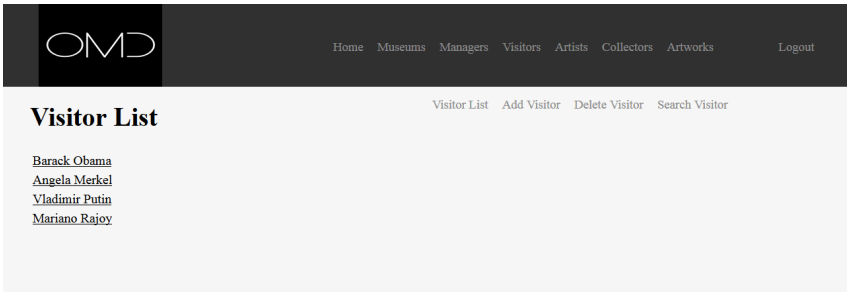


Figure 1.13: Figure 13: Visitor List Page

The visitor list on the page consists of links that directs the user to the display page of the selected visitor.

In the display page, the user can see the information of a visitor which are the name, age and nationality of the visitor. Also there is a list of the visited museums, which can be clicked to reach the display pages of those museums. Rest of the pages are pretty much the same with the managers and do not needed to be gone over again.

1.3 Parts Implemented by Tamer Temizer

1.3.1 Artworks

Artworks tab can be reached from the main navigation panel.

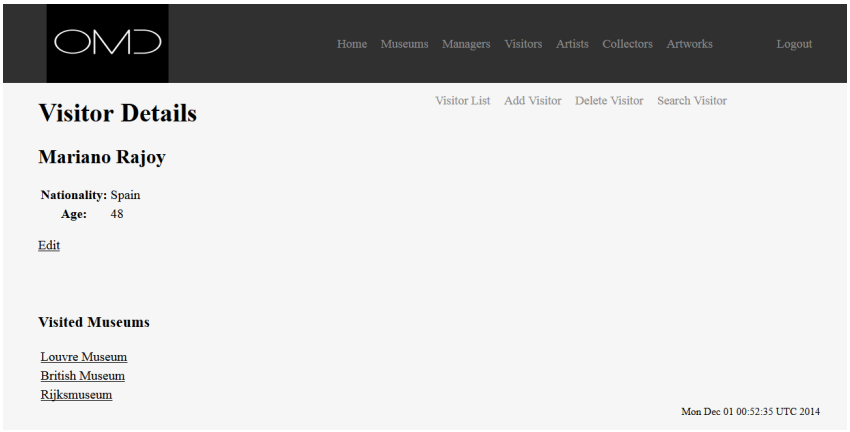
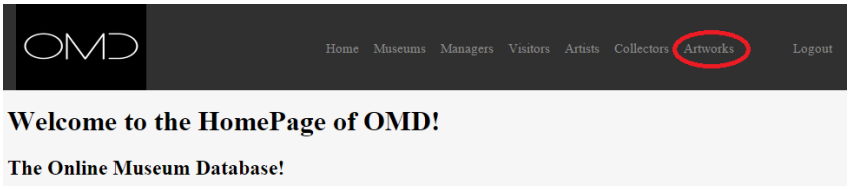
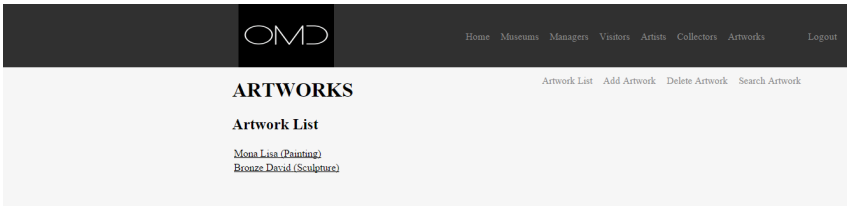


Figure 1.14: Figure 14: Visitor Display Page


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Welcome to the HomePage of OMD!
The Online Museum Database!

Main Page

This is the first page encountered when a user clicks 'Artworks' link on the panel. This page contains links to all artworks on the database. If a user clicks one of those links, display page of that artwork is opened.



Panel

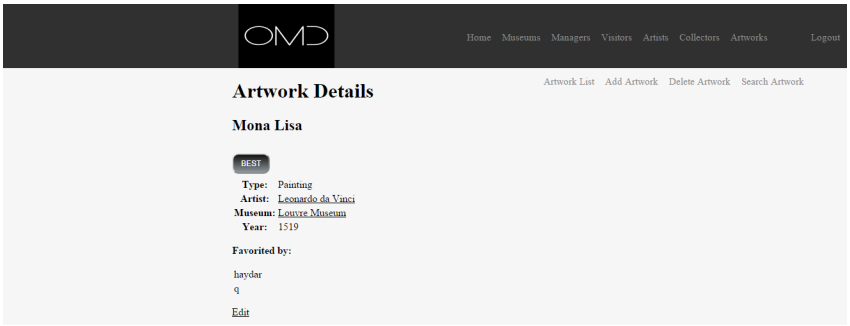
Artworks panel reacts differently if the user is an ordinary user or an editor. If the user is an ordinary user, he or she can only see 'Artwork List' and 'Search Artwork' links on the panel. If the user is an editor, he or she can see 'Add Artwork' and 'Delete Artwork' links too.

Display Page

If a user clicks one of the links on the artworks main page, this page opens. This page contains type, artist, museum and year of the clicked artwork. Artist and museum slots holds links to that particular artist's or museum's own page on the website. It also shows all the users who favorited this artwork. There is a button on the top of the page which allows the user to favorite it. There is a link at the bottom of the page, which can only be seen if the user is an editor, that allows to edit this artwork.

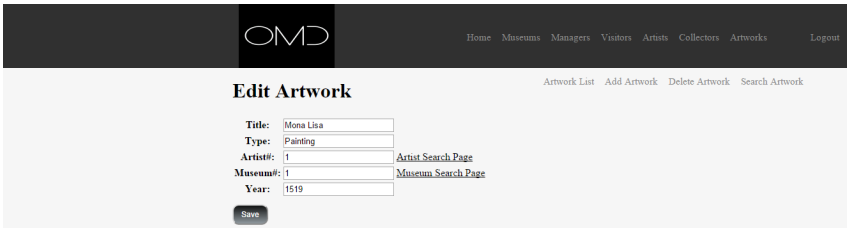
List Page

This link on the panel allows the user to return to the main page of artworks tab.



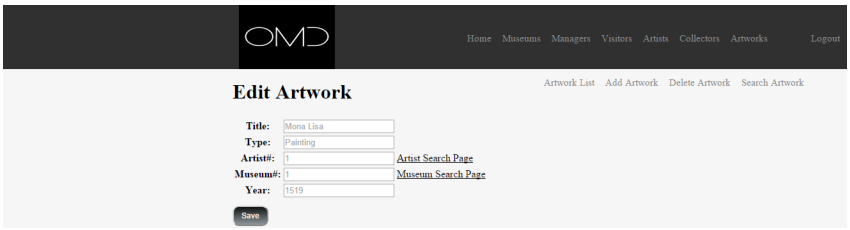
Update Page

If a user clicks the 'Edit' link on the display page, this page opens. In this page there are textboxes filled with the artwork's current information. The editor can change these values and save. There are also links to 'Artist Search Page' and 'Museum Search Page' to easily find a museum's or artist's id number as their id numbers are needed while editting.



Add Page

This page opens when an editor clicks 'Add Artwork' link on the panel. This page contains textboxes for title, type, artist id, museum id and year, like the update page. The editor can fill these textboxes and click save button to add a new artwork.

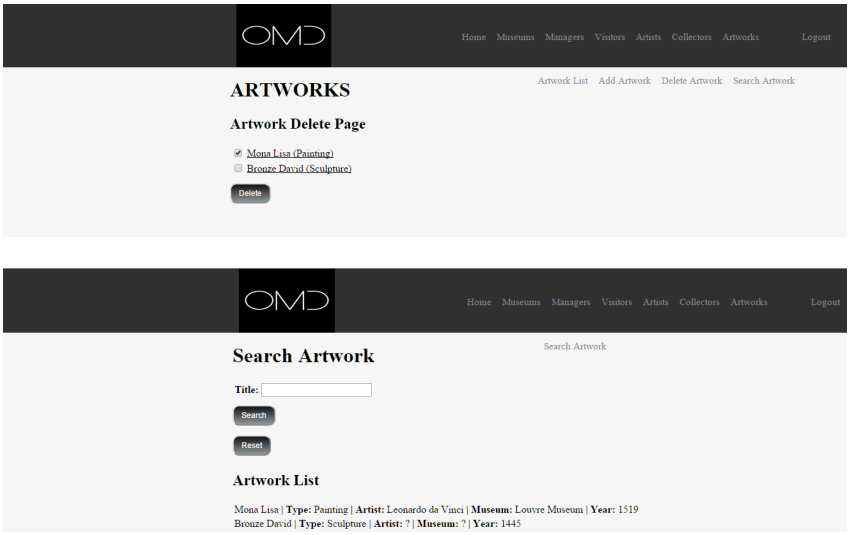


Delete Page

This page opens when an editor clicks 'Delete Artwork' link on the panel. This page contains the list of all artworks in the database with a checkbox next to their title. The editor can check these checkboxes and click delete button to delete checked artworks.

Search Page

This page opens when a user clicks 'Search Artwork' link on the panel. This is the only link on the panel that can be reached by ordinary users. In this page the user can type a text in the textbox and search for artworks starting with that text. If the artwork doesn't have a link to a museum or an artist, that slot is showed as '?'. If the user is an editor, he or she will also see the id of the artwork next to its title. The user can reset the search by clicking reset button. This will allow he or she to see the full list again.


OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
ARTWORKS
Artwork List Add Artwork Delete Artwork Search Artwork
Artwork Delete Page
Mona Lisa (Painting)
Bronze David (Sculpture)
Delete
OMD
Home Museums Managers Visitors Artists Collectors Artworks Logout
Search Artwork
Search Artwork
Title:
Search
Reset
Artwork List
Mona Lisa | Type: Painting | Artist: Leonardo da Vinci | Museum: Louvre Museum | Year: 1519
Bronze David | Type: Sculpture | Artist: ? | Museum: ? | Year: 1445

1.4 Parts Implemented by Haydar Şahin

1.4.1 Login & Signup

When am user enters to omd-dev.bluemix.net, he or she will firstly see a login screen.

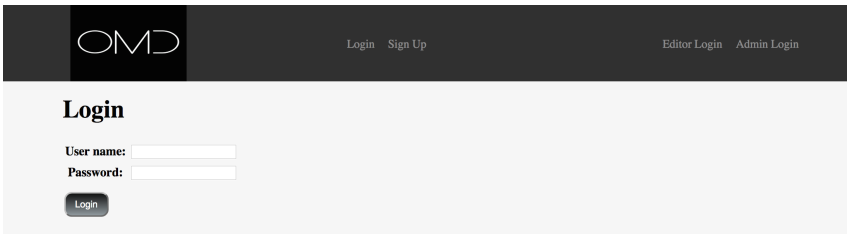


Figure 1.15: Figure.1: User Login Page

User

If user does have an account in the system, he or she can easily log in with his/her username and password. However, if user doesn't have any account on the system, the user should navigate to sign up page.

In the sign up page, user should fill up some information which is wanted by the website. These informations are: username, password, e-mail and birthday. After entering these information, the user can easily log in with his/her username and password form the login screen.

Editor

If the user is an editor, user should navigate to editor login page to log in as editor. No user can sign up as editor, editors are employees of omd-dev.bluemix.net.

Admin

If the user is an admin, user should navigate to admin login page to log in as admin. No user can signup as admin, only the admin with root privelege, which has a priority number 0, can only add or remove admins and editors.

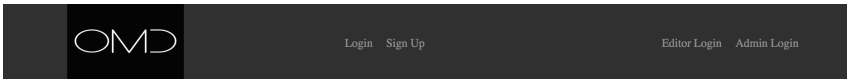


Figure 1.16: Figure.2: Signup Navigation Panel

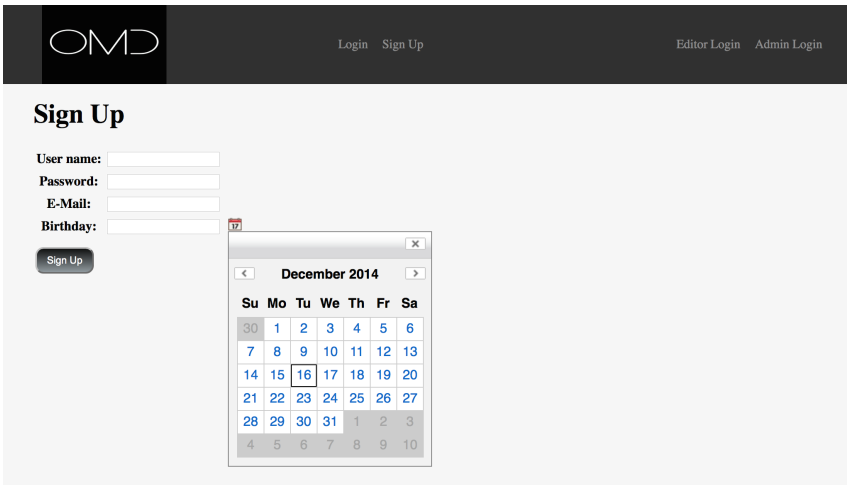


Figure 1.17: Figure.3: User Sign Up Page

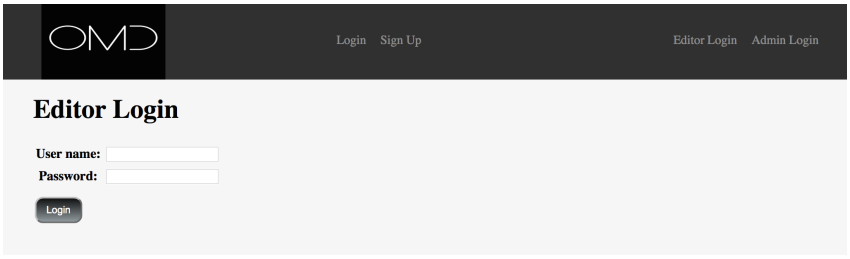


Figure 1.18: Figure.4: Editor Login Page

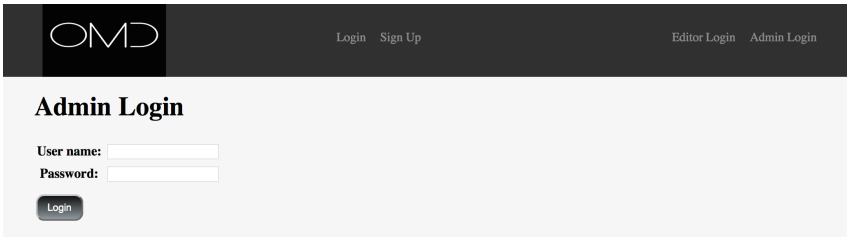


Figure 1.19: Figure.5: Admin Login Page

1.4.2 Users

After am user is logged in from the User Login Page, he/she will enter to homepage. In the homepage, there is an welcome screen. After that moment, user can select which topic he/she wants to get information from the navigation panel. The topics are Museums, Managers, Visitors, Artists, Collectors and Artworks. Moreover, the user can logout from the logout link from the navigation panel.

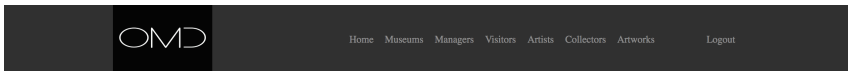


Figure 1.20: Figure.6: Main Navigation Panel

Interaction with Museums

In the museums page, an user can see the list of the museums and if he/she want to search museums; after selecting the 'Search Museum' link, he/she can search museums with their names from the search bar in the page. Users can see details of museums by clicking on their names in the list. After they click on the name of a specific museum, a detailed page of that museum will be shown to user. Moreover the user can make this museum his/her favourite museum by clicking 'Best' button.

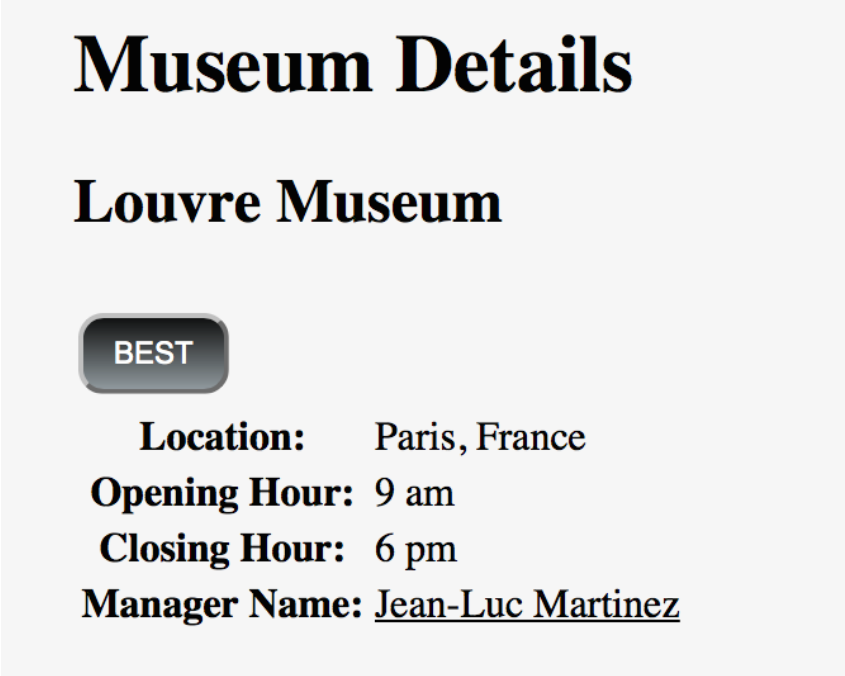


Figure 1.21: Figure.7: Museum Best Button

Detailed information is given under the Museums Page section of User Manual.

Interaction with Managers

In the managers page, an user can see the list of the managers and if he/she want to search managers; after selecting the 'Search Manager' link, he/she can search managers with their names from the search bar in the page. Users can see details of managers by clicking on their names in the list. After they click on the name of a specific manager, a detailed page of that manager will be shown to user. Detailed information is given under the Managers Page section of User Manual.

Interaction with Visitors

In the visitors page, an user can see the list of the visitors and if he/she want to search visitors; after selecting the 'Search Visitor' link, he/she can search visitors with their names from the search bar in the page. Users can see details of visitors by clicking on their names in the list. After they click on the name of a specific visitor, a detailed page of that visitor will be shown to user. Detailed information is given under the Visitors Page section of User Manual.

Interaction with Artists

In the artists page, an user can see the list of the artists and if he/she want to search artists; after selecting the 'Search Artist' link, he/she can search artists with their names from the search bar in the page. Users can see details of artists by clicking on their names in the list. After they click on the name of a specific artist, a detailed page of that artist will be shown to user. Moreover the user can make this artist his/her favourite artist by clicking 'Best' button.

Figure 1.22: Figure.8: Artist Best Button

Detailed information is given under the Artists Page section of User Manual.

Interaction with Collectors

In the collectors page, an user can see the list of the collectors and if he/she want to search collectors; after selecting the 'Search Collector' link, he/she can search collectors with their names from the search bar in the page. Users can see details of collectors by clicking on their names in the list. After they click on the name of a specific collector, a detailed page of that collector will be shown to user. Detailed information is given under the Collectors Page section of User Manual.

Interaction with Artworks

In the artworks page, an user can see the list of the artworks and if he/she want to search artworks; after selecting the 'Search Artwork' link, he/she can search artworks with their names from the search bar in the page. Users can see details of artworks by clicking on their names in the list. After they click on the name of a specific artwork, a detailed page of that artwork will be shown to user. Moreover the user can make this artwork his/her favourite artwork by clicking 'Best' button. Detailed information is given under the Artworks Page section of User Manual.

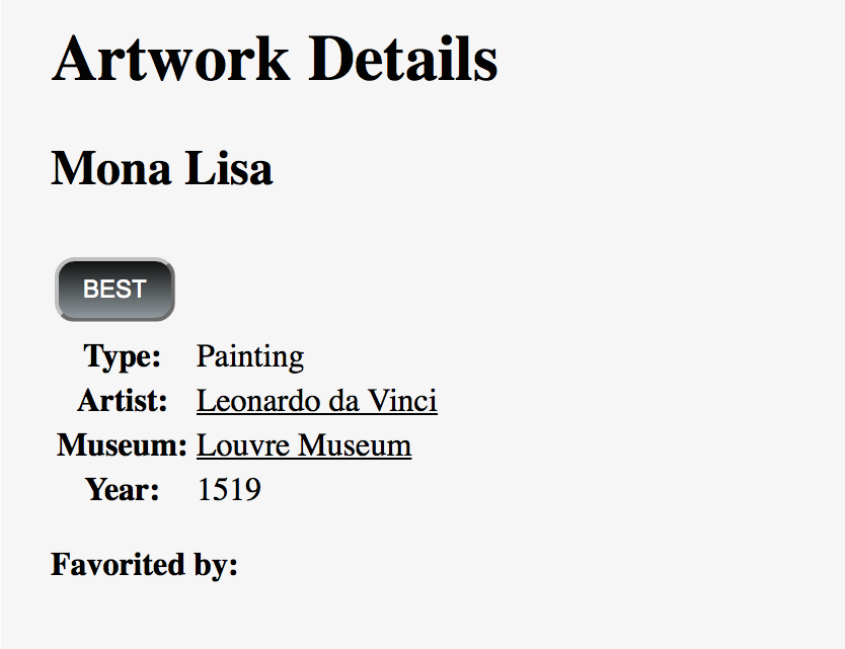


Figure 1.23: Figure.9: Artwork Best Button

1.4.3 Editors

After an editor is logged in from the Editor Login Page, he/she will enter to homepage. In the homepage, there is an welcome screen. After that moment, user can select which topic he/she wants to get information from the navigation panel. The topics are Museums, Managers, Visitors, Artists, Collectors and Artworks. Moreover, the user can logout from the logout link from the navigation panel.

Interaction with Museums

In the museums page, an editor can see the list of the museums and search museums like users. However, editors can add, remove and edit museums too. Detailed information is given under the Museums Page section of User Manual.

Interaction with Managers

In the managers page, an editor can see the list of the managers and search managers like users. However, editors can add, remove and edit managers too. Detailed information is given under the Managers Page section of User Manual.

Interaction with Visitors

In the visitors page, an editor can see the list of the visitors and search visitors like users. However, editors can add, remove and edit visitors too. Detailed information is given under the Visitors Page section of User Manual.

Interaction with Artists

In the artists page, an editor can see the list of the artists and search artists like users. However, editors can add, remove and edit artists too. Detailed information is given under the Artists Page section of User Manual.

Interaction with Collectors

In the collectors page, an editor can see the list of the collectors and search collectors like users. However, editors can add, remove and edit collectors too. Detailed information is given under the Collectors Page section of User Manual.

Interaction with Artworks

In the artworks page, an editor can see the list of the artworks and search artworks like users. However, editors can add, remove and edit artworks too. Detailed information is given under the Artworks Page section of User Manual.

1.4.4 Admins

After an user is logged in from the Admin Login Page, he/she will firstly see the user list. After that page the admin can select firstly what operation to do, and then select for which user class he/she want to do operation from the admin panel. There are four types of operations, which are 'List','Search','Insert','Delete' and 3 type of users 'User','Editor','Admin'.

Figure 1.24: Figure.10: Admin Panel to Select Operation

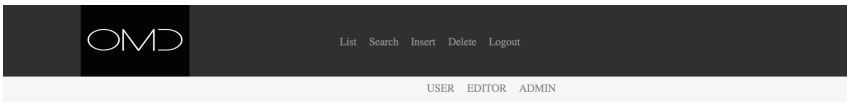


Figure 1.25: Figure.11: Admin Panel to Select User Type

There is two type of admins. One of them has root privileges. The others doesn't have root privileges. Admins with root privileges can see the password of users, editors and admins. However admins without root privileges can't see the password of users, editors and admins.

Interaction with Users

Admins can list, search, insert and remove users.

Moreover Admins can follow Users, however this is not an observable feature.

Interaction with Editors

Admins can list, search, insert and remove editors.

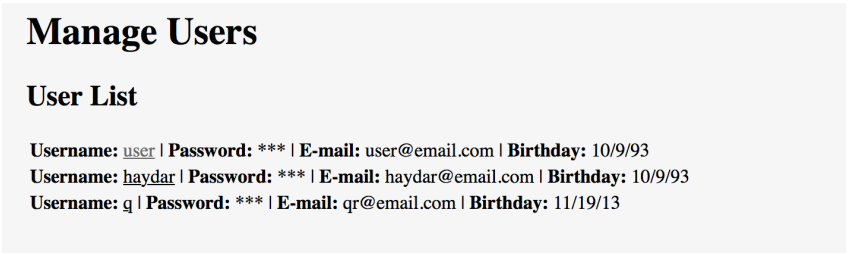

Manage Users
User List
Username: user | Password: *** | E-mail: user@email.com | Birthday: 10/9/93
Username: haydar | Password: *** | E-mail: haydar@email.com | Birthday: 10/9/93
Username: q | Password: *** | E-mail: qr@email.com | Birthday: 11/19/13

Figure 1.26: Figure.12: User List Page

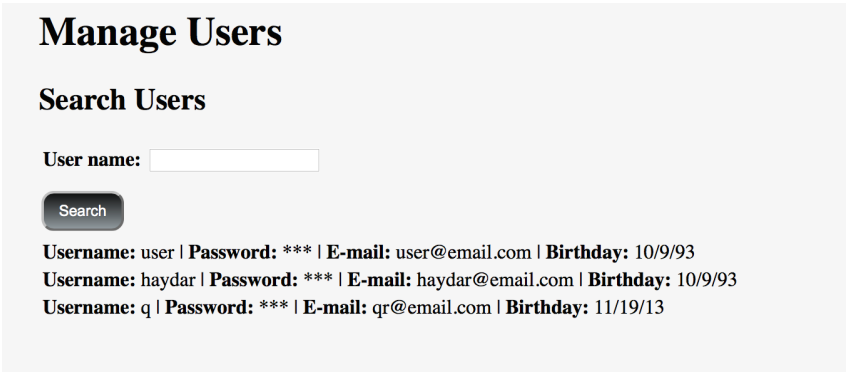

Manage Users
Search Users
User name:
Search
Username: user | Password: *** | E-mail: user@email.com | Birthday: 10/9/93
Username: haydar | Password: *** | E-mail: haydar@email.com | Birthday: 10/9/93
Username: q | Password: *** | E-mail: qr@email.com | Birthday: 11/19/13

Figure 1.27: Figure.13: User Search Page

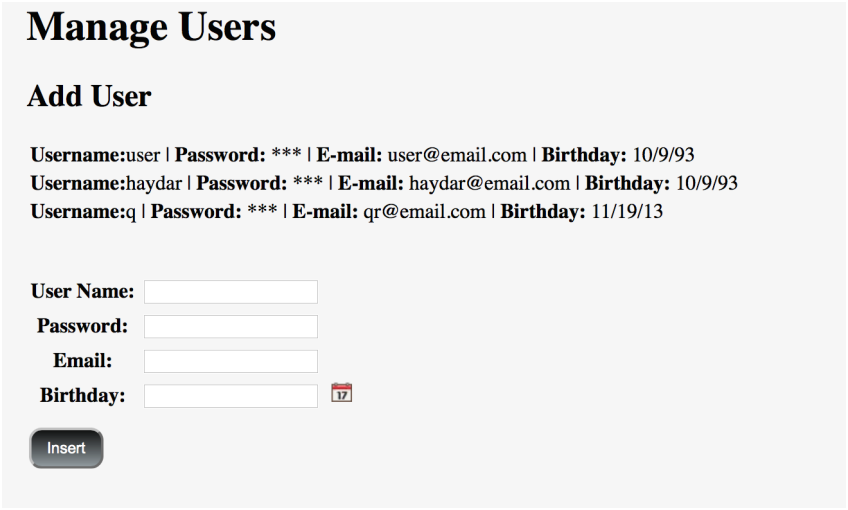

Manage Users
Add User
Username:user | Password: *** | E-mail: user@email.com | Birthday: 10/9/93
Username:haydar | Password: *** | E-mail: haydar@email.com | Birthday: 10/9/93
Username:q | Password: *** | E-mail: qr@email.com | Birthday: 11/19/13
User Name:
Password:
Email:
Birthday:
Insert

Figure 1.28: Figure.14: User Insert Page

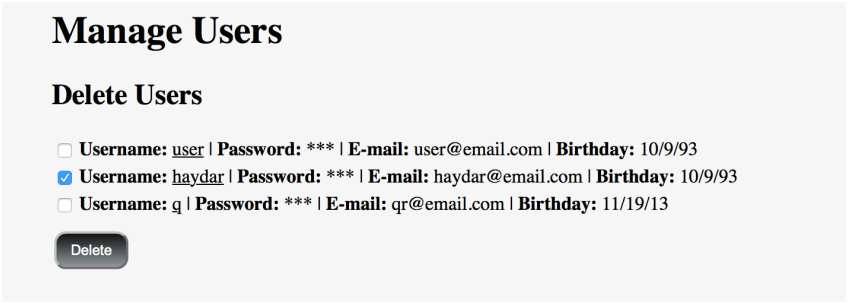

Manage Users
Delete Users
Username: user | Password: *** | E-mail: user@email.com | Birthday: 10/9/93
Username: haydar | Password: *** | E-mail: haydar@email.com | Birthday: 10/9/93
Username: q | Password: *** | E-mail: qr@email.com | Birthday: 11/19/13
Delete

Figure 1.29: Figure.15: User Delete Page

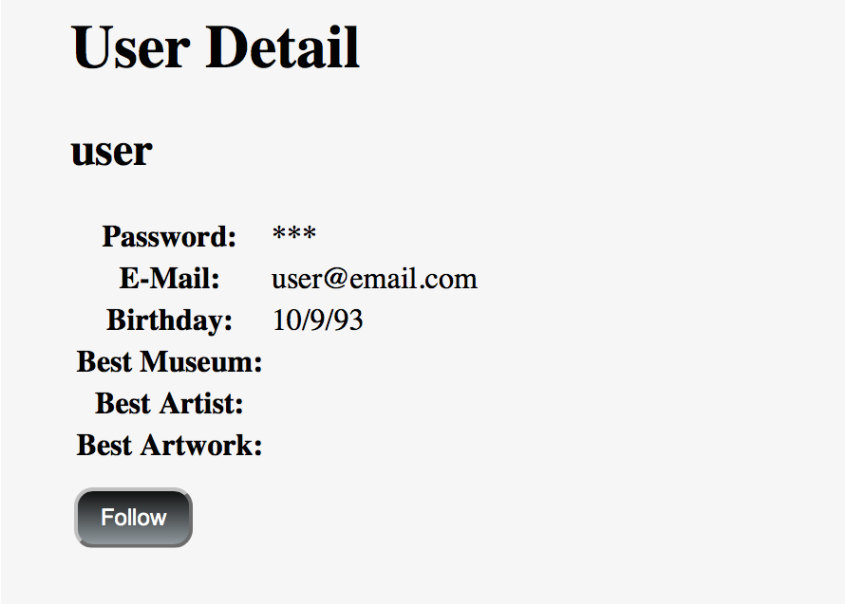


Figure 1.30: Figure.16: User Detail Page

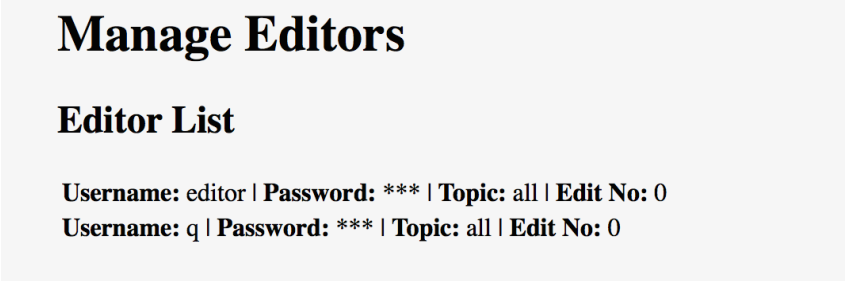


Figure 1.31: Figure.17: Editor List Page

Manage Editors
Search Editors
User name:
Search
Username: editor | Password: *** | Topic: all | Edit No: 0
Username: q | Password: *** | Topic: all | Edit No: 0

Figure 1.32: Figure.18: Editor Search Page

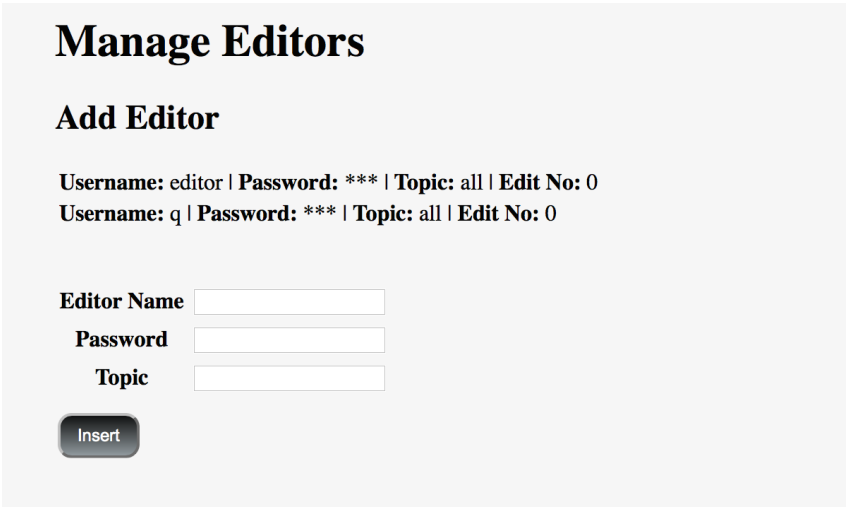

Manage Editors
Add Editor
Username: editor | Password: *** | Topic: all | Edit No: 0
Username: q | Password: *** | Topic: all | Edit No: 0
Editor Name
Password
Topic
Insert

Figure 1.33: Figure.19: Editor Insert Page

Manage Editors
Delete Editor
Username: editor | Password: *** | Topic: all | Edit No: 0
Username: q | Password: *** | Topic: all | Edit No: 0
Delete

Figure 1.34: Figure.20: Editor Delete Page

Interaction with Admins

Admins can list, search, insert and remove admins.

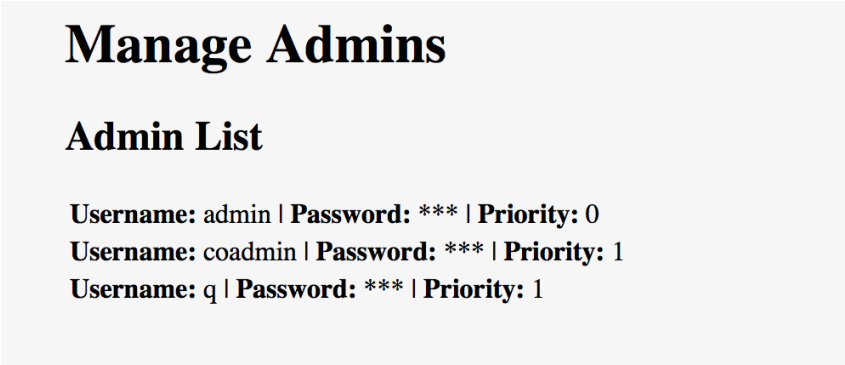

Manage Admins
Admin List
Username: admin | Password: *** | Priority: 0
Username: coadmin | Password: *** | Priority: 1
Username: q | Password: *** | Priority: 1

Figure 1.35: Figure.21: Admin List Page

Manage Admins
Search Admins
User name:
Search
Username: admin | Password: *** | Priority: 0
Username: coadmin | Password: *** | Priority: 1
Username: q | Password: *** | Priority: 1

Figure 1.36: Figure.22: Admin Search Page

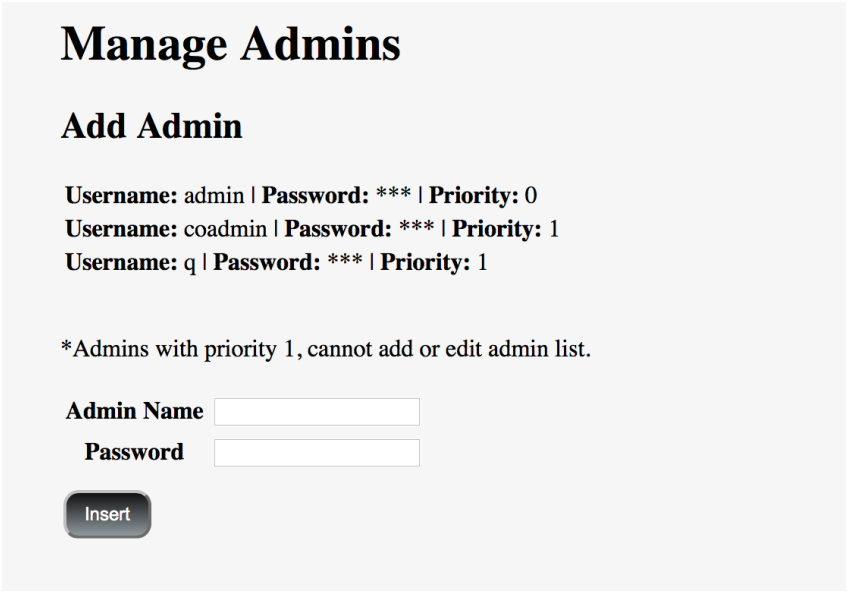

Manage Admins
Add Admin
Username: admin | Password: *** | Priority: 0
Username: coadmin | Password: *** | Priority: 1
Username: q | Password: *** | Priority: 1
*Admins with priority 1, cannot add or edit admin list.
Admin Name
Password
Insert

Figure 1.37: Figure.23: Admin Insert Page

Manage Admins
Delete Admin
*Admins with priority 1, can only delete itself.
Username: admin | Password: *** | Priority: 0
Username: coadmin | Password: *** | Priority: 1
Username: q | Password: *** | Priority: 1
Delete

Figure 1.38: Figure.24: Admin Delete Page

Developer Guide

Our Database does exist of 12 tables, which are:

- **Admin**
- **ARTIST**
- **ARTWORK**
- **COLLECTOR**
- **Editor**
- **EditorTopic**
- **FAVUSERS**
- **MANAGER**
- **MUSEUM**
- **User**
- **VISITING**
- **VISITOR**

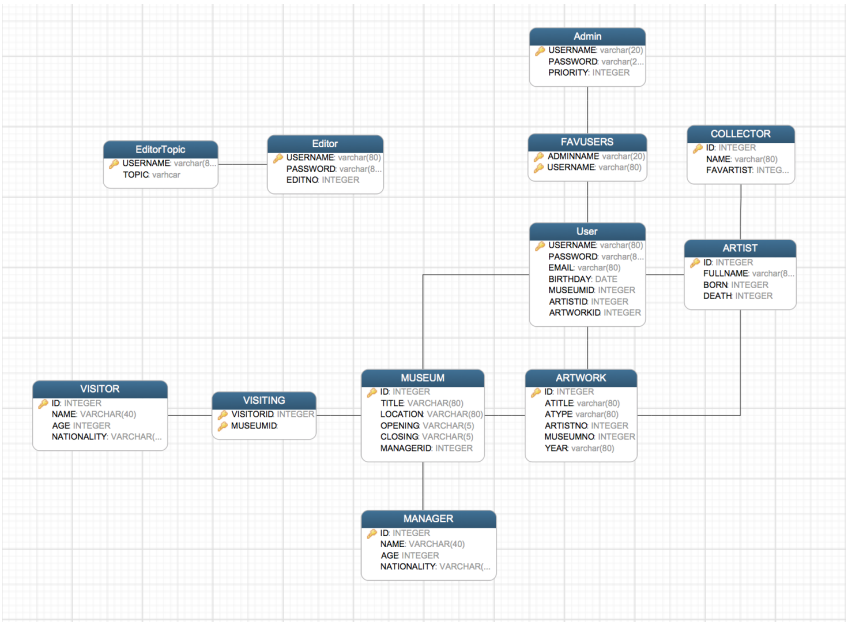


Figure 2.1: Figure.1: E/R Diagram of the database

2.1 Parts Implemented by Kerem Adalı

2.1.1 Database

Artist Table

```
CREATE TABLE ARTIST(
   ID INTEGER PRIMARY KEY AUTOINCREMENT,
   FULLNAME varchar(80) UNIQUE,
   BORN INTEGER,
   DEATH INTEGER
);
```

Artist table has 4 attributes. ID is the primary key value which is auto incremented when a new artist is added.

ID	FULLNAME	BORN	DEATH
1	Leonardo da Vinci	1452	1519
2	Pablo Picasso	1881	1973
3	Vincent Van Gogh	1853	1890
4	xxx	1853	1890

Collector Table

```
CREATE TABLE COLLECTOR(
   ID INTEGER PRIMARY KEY AUTOINCREMENT,
   NAME varchar(80) UNIQUE,
   FAVARTIST INTEGER REFERENCES ARTIST ON DELETE SET NULL
 );
```

Collector table has 3 attributes. ID is the primary key value which is auto incremented when a new collector is added.FAVARTIST is a primary key to the Artist Table which is set to NULL on a related deletion at Artist Table.

ID	NAME	FAVARTIST
1	John Smith	
2	Mr. Smith	

Entity / Relationship Diagram

- There is a 1:1 relationship between artist and collector.

2.1.2 Coding

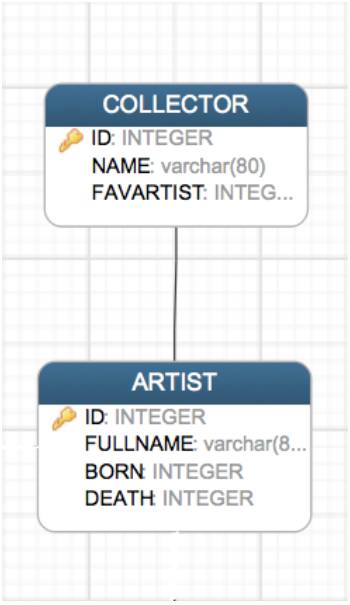
Artist

Artist is the class of the painters, sculptors etc.

Class

```
public class Artist {

    private Integer ID = null;
    private String fullname = null;
    private Integer born = null;
    private Integer death = null;
```




```
    //Methods
}
```

Artist class has 4 attributes which are ID, fullname, birth and death years.

Methods

Standart get/set methods.

Interface

```
public interface IArtistCollection {

    public List<Artist> getArtists();

    public void addArtist(Artist artist);

    public void deleteArtist(Artist artist);

    public void updateArtist(Artist artist);

    public Artist getArtist(Artist artist);

    public List<Artist> searchArtist(String search);

    public Artist getArtistWithID(Integer id);
}
```

Collection Methods

- **getArtists():** Gets the info about artists and returns a linked list of artists.
- **addArtist(Artist artist):** Adds artist to the artist table.
- **deleteArtist(Artist artist):** Deletes artist from the artist table.
- **updateArtist(Artist artist):** Updates artist variables at the table.

- **getArtist(Artist artist):** Searches artist by its name and returns a artist.
- **searchArtist(String search):** Searches artist by its name and returns a linked list of artists.
- **getArtistWithID(Integer id):** Searches artist by its ID and returns a artist.

Collector

Collector is the class for collectors at the database.

Class

```
public class Collector {

    private Integer ID = null;
    private String name = null;
    private Integer favartist = null;

    //Methods
 }
```

Collector class has 3 attributes which are ID, name and favartist(Id of the favorite artist).

Methods

Standart get/set methods and

```
public class Artist {

   //Attributes
   //Get/Set Methods

   public void setFavArtist(Integer afav_artist) {...}
   public Integer getFavArtist() {...}

}
```

- **setFavArtist:** Sets the favorite artist of the collector.
- **getFavArtist:** Returns the id of the favorite artist of that collector.

Interface

```
public interface ICollectorCollection {

    public List<Collector> getCollectors();

    public void addCollector(Collector collector);

    public void deleteCollector(Collector collector);

    public void updateCollector(Collector collector);

    public Collector getCollector(Collector collector);

    public List<Collector> searchCollector(String search);

    public Collector getCollectorWithID(Integer id);
```

```
}
```

Collection Methods

- **getCollectors():** Gets the info about collectors and returns a linked list of collectors.
- **addCollector(Collector collector):** Adds collector to the collector table.
- **deleteCollector(Collector collector):** Deletes collector from the collector table.
- **updateCollector(Collector collector):** Updates collector variables at the table.
- **getCollector(Collector collector):** Searches collector by its name and returns a collector.
- **searchCollector(String search):** Searches collector by its name and returns a linked list of collectors.
- **getCollectorWithID(Integer id):** Searches collector by its ID and returns a collector.

2.2 Parts Implemented by Arda Özdere

2.2.1 Database Design

MUSEUM Table

```
CREATE TABLE MUSEUM(
ID INTEGER PRIMARY KEY NOT NULL,
TITLE VARCHAR(80) UNIQUE,
LOCATION VARCHAR(80),
OPENING VARCHAR(5),
CLOSING VARCHAR(5),
MANAGERID INTEGER REFERENCES MANAGER ON DELETE SET NULL);
```

Museum table has 6 attributes. Id is the primary key which can not take a null value and auto incremented when a new museum is added. Title is the name of the museum, location is the location of the museum. Opening and closing hours are restricted by 5 characters. Manager id is a foreign key to the MANAGER table, which is set to null on a deletion.

ID	TITLE	LOCATION	OPENING	CLOSING	MANAGERID
1	Louvre Museum	Paris, France	9 am	6 pm	1
2	British Museum	London, England	10 am	7 pm	2
3	Rijksmuseum	Amsterdam, Neth...	10 am	6 pm	3
4	GemaldeGalerie	Berlin, Germany	9 am	6 pm	4

Figure 2.2: Museum Table

MANAGER Table

```
CREATE TABLE MANAGER(
ID INTEGER PRIMARY KEY NOT NULL,
NAME VARCHAR(40),
AGE INTEGER,
NATIONALITY VARCHAR(20));
```

Manager table has 4 attributes. The id of the manager is the primary key which can not be null and is auto incremented in each addition. Also the name, age and nationality of the managers are included.

ID	NAME	AGE	NATIONALITY
1	Jean-Luc Martinez	57	France
2	Neil MacGregor	68	Scotland
3	Hoop Scheffer	62	Netherlands
4	Michael Eissenhauer	49	Germany

Figure 2.3: Manager Table

VISITOR Table

```
CREATE TABLE VISITOR(
ID INTEGER PRIMARY KEY NOT NULL,
NAME VARCHAR(40),
AGE INTEGER,
NATIONALITY VARCHAR(20));
```

Visitor table has 4 attributes. The id of the visitor is the primary key which can not be null and is auto incremented in each addition. Also the name, age and nationality of the visitors are included.

ID	NAME	AGE	NATIONALITY
1	Barack Obama	53	USA
2	Angela Merkel	57	Germany
3	Vladimir Putin	55	Russia
4	Mariano Rajoy	48	Spain

Figure 2.4: Visitor Table

VISITING Table

```
CREATE TABLE VISITING(
VISITORID INTEGER REFERENCES VISITOR ON DELETE CASCADE,
MUSEUMID INTEGER REFERENCES MUSEUM ON DELETE CASCADE,
PRIMARY KEY ( VISITORID, MUSEUMID));
```

Visiting table has 2 attributes. These are both foreign keys. One references the visitor table and one references the museum table. Both are set to cascade on any deletions. The primary key is a combination of both of these foreign keys.

Entity / Relationship Diagram

Relationships

- Museum-Manager : 1 to 1
- Museum-Artwork : 1 to 1
- Museum-User : 1 to 1
- Museum-Visiting : n to n
- Visitor-Visiting : n to n
- Visitor-Museum : 1 to n

VISITORID	MUSEUMID
1	1
2	3
2	4
2	1
2	2
3	2
3	1
4	1
4	2
4	3

Figure 2.5: Visiting Table

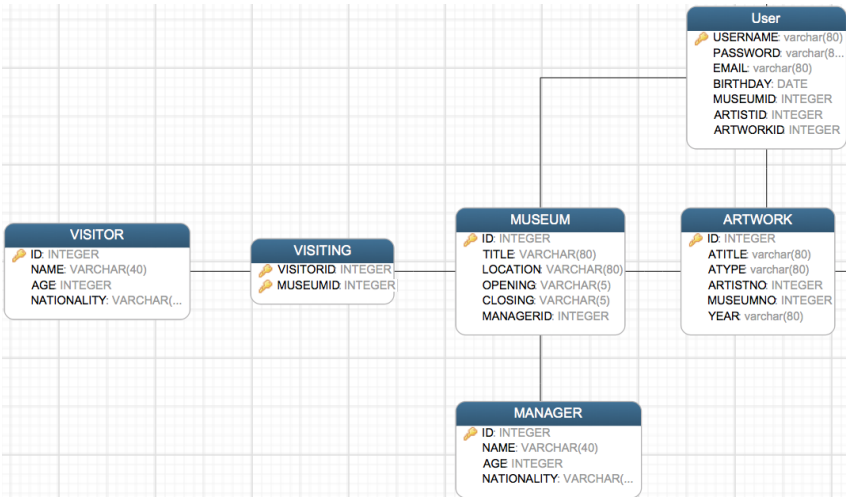


Figure 2.6: E/R Diagram

2.2.2 Code

Museum Class

```
public class Museum {

private String title = null;
private String location = null;
private String openingHour = null;
private String closingHour = null;
private Integer id = null;
private Integer managerid = null;

//getters and setters

}
```

- **title:** Name of the museum
- **id:** ID of the museum
- **openingHour:** Opening hour of the museum
- **closingHour:** Closing hour of the museum
- **location:** Location of the museum
- **managerid:** Manager ID of the museum (Foreign key)

Museum JDBC Operations

The query of the operations are commented below the Java functions.

```
public interface IMuseumCollection {

public List<Museum> getMuseums();
//"SELECT * FROM MUSEUM"

public void addMuseum(Museum museum);
//"INSERT INTO MUSEUM (TITLE, LOCATION, OPENING, CLOSING, MANAGERID)
//                      VALUES (?, ?, ?, ?, ?)"

public void updateMuseum(Museum museum);
//"UPDATE MUSEUM SET TITLE = ?, LOCATION = ?, OPENING = ?,
//                      CLOSING = ?, MANAGERID = ? WHERE (ID = ?)"

public void deleteMuseum(Museum museum);
//"DELETE FROM MUSEUM WHERE (ID = ?)"

public Museum getMuseum(Museum museum);
//"SELECT * FROM MUSEUM WHERE TITLE = '" + museum.getTitle() + "'"

public Museum getMuseumWithID(Integer id);
//"SELECT * FROM MUSEUM WHERE (ID = ?)"

public List<Museum> searchMuseums(String search);
//"SELECT * FROM MUSEUM WHERE TITLE LIKE '" + search + "%'"

}
```

- **getMuseums():** Returns a linked list of all museums.
- **addMuseum(Museum museum):** Adds a museum to the museum table.

- **deleteMuseum(Museum museum):** Deletes the desired museum from the museum table.
- **updateMuseum(Museum museum):** Updates the information of a museum with its new values.
- **getMuseum(Museum museum):** Searches a museum by its name and returns it.
- **searchMuseum(String search):** Searches a museum with a string like its name and returns a linked list of museums.
- **getMuseumWithID(Integer id):** Searches a museum by its ID and returns it.

Manager Class

```
public class Manager {
private String name = null;
private String nationality = null;
private Integer age = null;
private Integer id = null;

//getters and setters

}
```

- **name:** Name of the manager
- **nationality:** Nationality of the manager
- **age:** Age of the manager
- **id:** ID of the manager

Manager JDBC Operations

The query of the operations are commented below the Java functions.

```
public interface IManagerCollection {

public List<Manager> getManagers();
//"SELECT * FROM MANAGER"

public void addManager(Manager manager);
//"INSERT INTO MANAGER (NAME, AGE, NATIONALITY) VALUES (?, ?, ?)"

public void updateManager(Manager manager);
//"UPDATE MANAGER SET NAME = ?, AGE = ?, NATIONALITY = ? WHERE (ID = ?)"

public void deleteManager(Manager manager);
//"DELETE FROM MANAGER WHERE (ID = ?)"

public Manager getManager(Manager manager);
//"SELECT * FROM MANAGER WHERE NAME = '" + manager.getName() + "'"

public Manager getManagerWithID(Integer id);
//"SELECT * FROM MANAGER WHERE (ID = ?)"

public List<Manager> searchManagers(String search);
//"SELECT * FROM MANAGER WHERE NAME LIKE '" + search + "%'"

}
```

- **getManagers():** Returns a list of all managers.
- **addManager(Manager manager):** Adds a manager to the manager table.

- **deleteManager(Manager manager):** Deletes the desired manager from the manager table.
- **updateManager(Manager manager):** Updates the information of a manager with its new values.
- **getManager(Manager manager):** Searches a manager by its name and returns it.
- **searchManager(String search):** Searches a manager with a string like its name and returns a list of managers.
- **getManagerWithID(Integer id):** Searches a manager by its ID and returns it.

Visitor Class

```
public class Visitor {
private String name = null;
private String nationality = null;
private Integer age = null;
private Integer id = null;

//getters and setters

}
```

- **name:** Name of the visitor
- **nationality:** Nationality of the visitor
- **age:** Age of the visitor
- **id:** ID of the visitor

Visitor JDBC Operations

The query of the operations are commented below the Java functions.

```
public interface IVisitorCollection {

public List<Visitor> getVisitors();
//"SELECT * FROM VISITOR"

public void addVisitor(Visitor visitor);
//"INSERT INTO VISITOR (NAME, AGE, NATIONALITY) VALUES (?, ?, ?)"

public void updateVisitor(Visitor visitor);
//"UPDATE VISITOR SET NAME = ?, AGE = ?, NATIONALITY = ? WHERE (ID = ?)"

public void deleteVisitor(Visitor visitor);
//"DELETE FROM VISITOR WHERE (ID = ?)"

public Visitor getVisitor(Visitor visitor);
//"SELECT * FROM VISITOR WHERE NAME = '"

public Visitor getVisitorWithID(Integer id);
//"SELECT * FROM VISITOR WHERE (ID = ?)"

public List<Visitor> searchVisitors(String search);
//"SELECT * FROM VISITOR WHERE NAME LIKE '" + search + "%'"

public List<Museum> searchVisitedMuseumsWithVisitorId(Integer VisitorId);
//"SELECT * FROM MUSEUM, VISITING, VISITOR WHERE ( (VISITORID=VISITOR.ID) AND
//          (MUSEUMID=MUSEUM.ID) AND (VISITORID = " + VisitorId + "))"

}
```

- **getVisitors():** Returns a list of all visitors.
- **addVisitor(Visitor visitor):** Adds a visitor to the visitor table.
- **deleteVisitor(Visitor visitor):** Deletes the desired visitor from the visitor table.
- **updateVisitor(Visitor visitor):** Updates the information of a visitor with its new values.
- **getVisitor(Visitor visitor):** Searches a visitor by its name and returns it.
- **searchVisitor(String search):** Searches a visitor with a string like its name and returns a list of visitors.
- **getVisitorWithID(Integer id):** Searches a visitor by its ID and returns it.
- **searchVisitedMuseumsWithVisitorId(Integer VisitorId):** Returns a list of the visited museums of the visitor with the given ID.

2.3 Parts Implemented by Tamer Temizer

2.3.1 Database Design

ARTWORK Table

```
CREATE TABLE ARTWORK(
ID INTEGER PRIMARY KEY AUTOINCREMENT,
ATITLE varchar(80),
ATYPE varchar(80),
ARTISTNO  INTEGER REFERENCES ARTIST ON DELETE SET NULL,
MUSEUMNO INTEGER REFERENCES MUSEUM ON DELETE SET NULL,
YEAR varchar(80));
```

Artwork table has 6 attributes. ID is the primary key which can't be null and increments when a new artwork is added. ATITLE is the title of the artwork it is a varchar with 80 characters (Mona Lisa, Starry Night etc.). ATYPE is the type of the artwork it is a varchar with 80 characters (painting, sculpture etc.). MUSEUMNO and ARTISTNO are integer foreign keys that references MUSEUM table and ARTIST tables respectively. They hold IDs of a museum and artist and they are used to create links to that page. If the stored artist or museum is deleted, ARTISTNO or MUSEUMNO becomes NULL. YEAR is the year artwork was created, it is a varchar with 80 characters. YEAR isn't an integer because creation year can be an interval or unknown.

ID	ATITLE	ATYPE	ARTISTNO	MUSEUMNO	YEAR
1	Mona Lisa	Painting	1	1	1519
2	Bronze David	Sculpture			1440

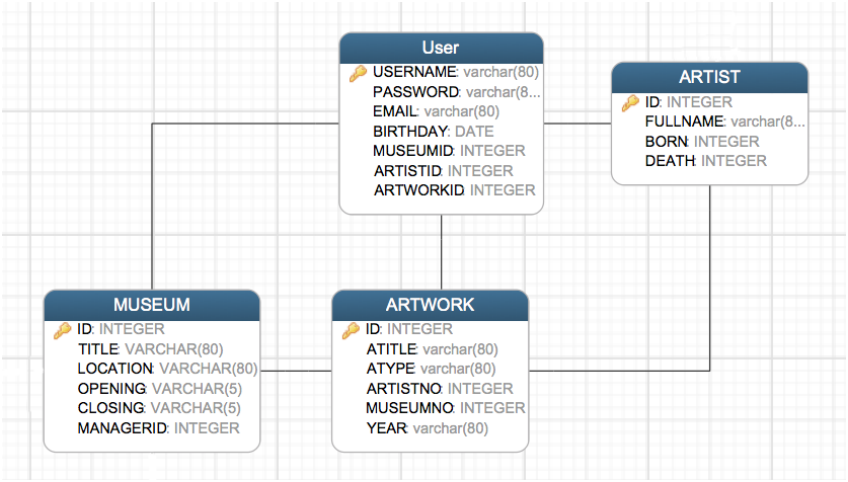
Entity / Relationship Diagram

Relationships

- Artwork - Museum : 1 to 1
- Artwork - Artist : 1 to 1
- Artwork - User : 1 to n

2.3.2 Code

Artwork Class




```
public class Artwork {

 private String title = null;
 private String type = null;
 private Integer artist = null;
 private Integer museum = null;
 private String year = null;
 private Integer ID = null;

 }
```

Getters and setters are omitted as they are trivial.

- **title:** Title of the artwork.
- **type:** Type of the artwork.
- **artist:** Creator of the artwork. It is a foreign key that references ARTIST table.
- **museum:** Museum of the artwork. It is a foreign key that references MUSEUM table.
- **year:** Year this artwork was created.
- **id:** ID of the artwork. This is unique and used as a primary key in the ARTWORK table.

Artwork JDBC Operations

The query of the operations can be seen below the Java functions.

```
public interface IArtworkCollection {

    public List<Artwork> getArtworks();
    //SELECT * FROM ARTWORK

    public List<String> getFavorited(Integer id);
    //SELECT * FROM User WHERE (ARTWORKID = ?)

    public void addArtwork(Artwork anArtwork);
    //INSERT INTO ARTWORK (ATITLE, ATYPE, ARTISTNO, MUSEUMNO, YEAR ) VALUES (?, ?, ?, ?, ?)

    public void updateArtwork(Artwork anArtwork);
    //UPDATE ARTWORK SET ATITLE = ?, ATYPE = ?, ARTISTNO = ?, MUSEUMNO = ?, YEAR = ? WHERE (ID =

    public void deleteArtwork(Artwork anArtwork);
    //DELETE FROM ARTWORK WHERE (ID = ?)
```

```
    public List<Artwork> searchArtworks(String search);
    //SELECT * FROM ARTWORK WHERE ATITLE LIKE '" + search + "%'"

    public Artwork getArtworkWithID(Integer id);
    //SELECT * FROM ARTWORK WHERE (ID = ?)
}
```

Usage of Artwork JDBC Operations

- **getArtworks():** Returns all artworks in the ARTWORK table. Used in listing all artworks.
- **getFavorited(Integer id):** Returns all users who favorited the artwork with given id. Used in display page to list all users favorited that artwork.
- **addArtwork(Artwork anArtwork):** Adds the given artwork to ARTWORK table. Used in 'Add Artwork' page.
- **updateArtwork(Artwork anArtwork):** Updates an artwork. Used in update page that is reached by clicking edit on display page.
- **deleteArtwork(Artwork anArtwork):** Deletes the given artwork from ARTWORK table. Used in 'Delete Artwork' page.
- **searchArtworks(String search):** Lists all artworks starting with the given string. Used in 'Search Artwork' page.
- **getArtworkWithID(Integer id):** Returns the artwork with the given ID. Used for cooperation with other tables.

2.4 Parts Implemented by Haydar Şahin

2.4.1 Database Design

There are three types of users in our website, and we have 5 tables in according to store these type of users and some of their attributes.

User

User Table

User Table stores information of user accounts.

Structure of User Table

USERNAME	VARCHAR(80)
PASSWORD	VARCHAR(80)
EMAIL	VARCHAR(80)
BIRTHDAY	DATE
MUSEUMID	INTEGER
ARTISTID	INTEGER
ARTWORKID	INTEGER

Figure 2.7: Figure.1: Structure of User Table

USERNAME is primary key.

MUSEUMID, **ARTISTID**, **ARTWORKID** are *foreign keys* to **MUSEUM**, **ARTIST**, **ARTWORK** tables. This relation is needed for favourite museum, artist and artwork feature of users.

```
CREATE TABLE User(
USERNAME varchar(80) PRIMARY KEY,
PASSWORD varchar(80) NOT NULL,
EMAIL varchar(80) NOT NULL,
BIRTHDAY DATE NOT NULL,
MUSEUMID INTEGER REFERENCES MUSEUM ON DELETE SET NULL,
ARTISTID INTEGER REFERENCES ARTIST ON DELETE SET NULL,
ARTWORKID INTEGER REFERENCES ARTWORK ON DELETE SET NULL)
```

MUSEUMID, **ARTISTID**, **ARTWORKID** has attribute *ON DELETE SET NULL*, because when an user has a museum/artist/artwork as favourite museum/artist/artwork, that museum/artist/artwork can be deleted.

E/R Diagram

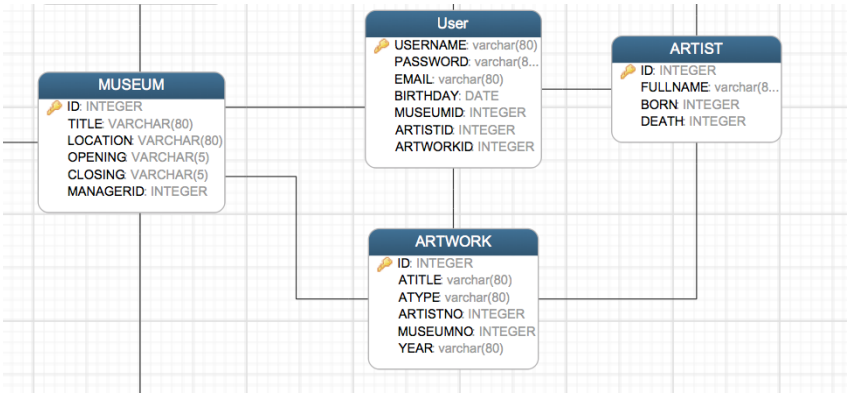


Figure 2.8: Figure.2: User Table E/R Diagram with MUSEUM, ARTIST and ARTWORK Tables.

All of these three foreign keys, are 'SET TO NULL ON DELETE'. With this feature, we disable any problem which can be occured by deleting museums, artist or artworks.

All Users can have only one favourited museum, artist or artwork. This will create a *1-n relation* between User Table and MUSEUM, ARTIST and ARTWORK tables.

Editor

Editor Table

Editor Table stores information of editor accounts.

Structure of Editor Table

USERNAME	VARCHAR(80)	
PASSWORD	VARCHAR(80)	
EDITNO	INTEGER	

Figure 2.9: Figure.3: Structure of Editor Table

USERNAME is primary key.

```
CREATE TABLE Editor(
USERNAME varchar(80) PRIMARY KEY,
PASSWORD varchar(80) NOT NULL,
EDITNO INTEGER DEFAULT 0)
```

EDITNO stores to total edit number of editors. That's why it is set to 0 as default. However after each edit, this value will increment.

EditorTopic Table

Editors do have topic/topics to be responsible of. This table stores the responsible topics of editors.

Structure of EditorTopic Table

USERNAME	VARCHAR(80)	
TOPIC	VARHCAR(80)	

Figure 2.10: Figure.4: Structure of EditorTopic Table

USERNAME is both primary and foreign key.

```
CREATE TABLE EditorTopic(
USERNAME varchar(80) REFERENCES Editor ON DELETE CASCADE,
TOPIC varhcar(80),
PRIMARY KEY ( USERNAME ))
```

USERNAME is *ON DELETE CASCADE*, because if when an editor is removed from Editor Table, the same editor must be deleted from EditorTopic table.

E/R Diagram

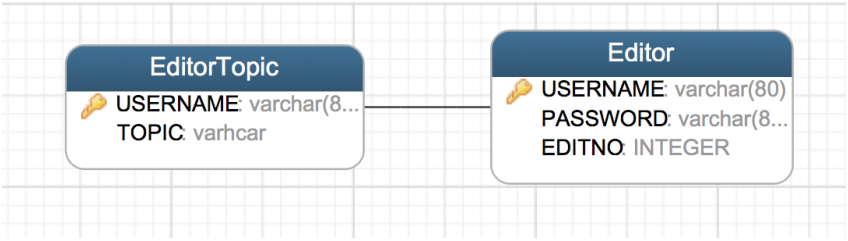


Figure 2.11: Figure.5: Editor E/R Diagram with Editor Table and EditorTopic Table.

Every editor can have only one topic, which are specific to them. This will create a *1-1 relation* between Editor table and EditorTopic Table.

Admin

Admin Table

Admin Table stores information of admin accounts.

Structure of Admin Table

USERNAME	VARCHAR(20)	
PASSWORD	VARCHAR(20)	
PRIORITY	INTEGER	

Figure 2.12: Figure.6: Structure of Admin Table

USERNAME is the primary key.

PRIORITY is just an integer value, which shows the privilege of admins. 0 means root privilege, 1 means normal privilege.

FAVUSERS Table

FAVUSERS Table stores the favourited users of admins.

Structure of FAVUSERS Table

ADMINNAME	VARCHAR(20)
USERNAME	VARCHAR(80)

Figure 2.13: Figure.7: Structure of FAVUSERS Table

E/R Diagram

Every admin can favourite more than one users and one user can be favourited by more than one admin. This will create a *n-n relation* between Admin table and FAVUSERS Table.

2.4.2 Code

Base User

Base User Class

Base User Class is the super class of the other user classes. This class is needed because we store the user, which has logged in. However if we think that there are three types of users, the best way to store these users in one variable is creating a super class for these users.

Base User Class doesn't have any table, however it is implemented in Java Code.

```
public class BaseUser {
        private String username = null;
        private String password = null;
}
```

BaseUser class just has two attributes, which are username and password.

User

User Class

User Class is the class which is implemented for ordinary users.

User Class inherits BaseUser class, and has more attributes then BaseUser class.

```
public class User extends BaseUser {
                private String email = null;
                private Integer bestArtist = null;
            private Integer bestMuseum = null;
            private Integer bestArtwork = null;
            private String birthday = null;
}
```

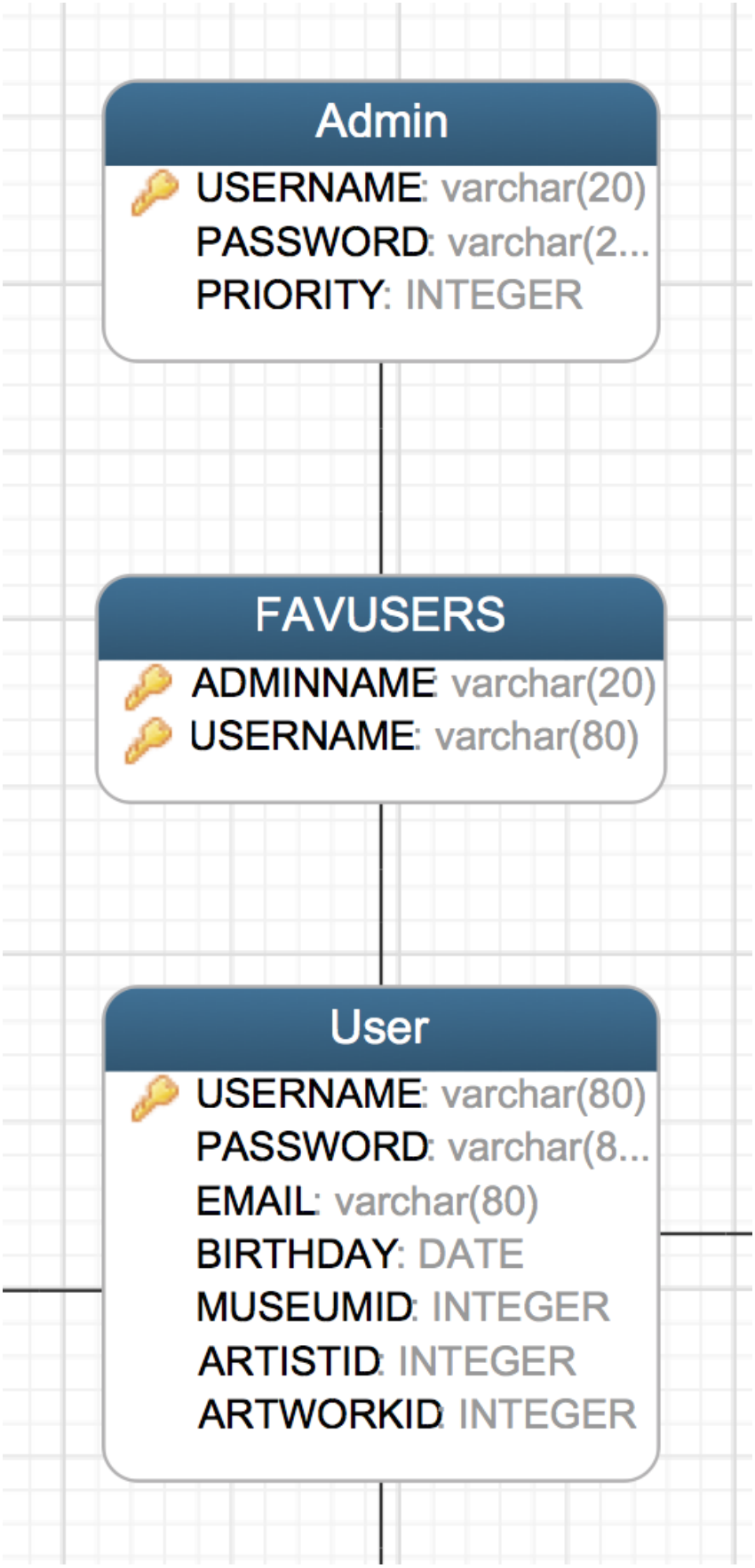


Figure 2.14: Figure.8: Admin E/R Diagram with Admin Table and FAVUSERS Table.

User JDBC Codes

There are 10 functions, which are query for User Table.

```
public interface IUserCollection {
        public List<User> getUsers();

        public void addUser(User user);

        public void deleteUser(User user);

        public List<User> searchUsers(String search);

        public void updateArtist(User user, Artist artist);

        public void updateMuseum(User user, Museum museum);

        public void updateArtwork(User user, Artwork artwork);

        public Boolean existsUser(User user);

        public Boolean existsUsername(User user);

        public User getUser(User user);
}
```

- **public List<User> getUsers():** Returns the whole list of Users with a simple *SELECT* query.
- **public void addUser(User user):** Inserts the given User to the User Table with a simple *INSERT INTO* query.
- **public void deleteUser(User user):** Removes the given User from the User table with a simple *DELETE* query.
- **public List<User> searchUsers(String search):** Searches users, whose name starts with the given string, with a simple *SELECT* query.
- **public void updateArtist(User user, Artist artist):** Updates the ARTISTID column of User Table.
- **public void updateMuseum(User user, Museum museum);:** Updates the MUSEUMID column of User Table.
- **public void updateArtwork(User user, Artwork artwork):** Updates the ARTWORKID column of User Table.
- **public Boolean existsUser(User user):** Returns '1' if there is an user with the given users name and password, otherwise returns '0'.
- **public Boolean existsUsername(User user):** Returns '1' if there is an user with the given users name, otherwise returns '0'.
- **public User getUser(User user):** Returns the information of User, which is sent to this function as parameter.

Editor

Editor Class

Editor Class is the class which is implemented for ordinary users.

Editor Class inherits BaseUser class, and has more attributes then BaseUser class.

```
public class Editor extends BaseUser {
        private Integer totaleditnumber = 0;
```

```
        private String topic;
}
```

Editor JDBC Codes

There are 7 functions, which are query for Editor Table.

```
public interface IEditorCollection {

        public List<Editor> getEditors();

        public void addEditor(Editor editor);

        public void deleteEditor(Editor editor);

        public List<Editor> searchEditors(String search);

        public Boolean existsEditor(Editor editor);

        public Editor getEditor(Editor editor);

        public void increase(Editor editor);

}
```

- **public List<Editor> getEditors():** Returns the whole list of Editors with a simple *SELECT* query.
- **public void addEditor(Editor editor):** Inserts the given Editor to the Editor Table with a simple *INSERT INTO* query.
- **public void deleteEditor(Editor editor):** Removes the given Editor from the Editor table with a simple *DELETE* query.
- **public List<Editor> searchEditors(String search):** Searches editors, whose name starts with the given string, with a simple *SELECT* query.
- **public Boolean existsEditor(Editor editor):** Returns '1' if there is an editor with the given editors name and password, otherwise returns '0'.
- **public Editor getEditor(Editor editor);:** Returns the information of Editor, which is sent to this function as parameter.
- **public void increase(Editor editor):** Increases the *EDITNO* column of Editor Table by one.

Admin

Admin Class

Editor Class is the class which is implemented for ordinary users.

Editor Class inherits BaseUser class, and has more attributes then BaseUser class.

```
public class Admin extends BaseUser {
        private Integer priority = null;
}
```

Admin JDBC Codes

There are 7 functions, which are query for Admin Table.

```
public interface IAdminCollection {

        public List<Admin> getAdmins();

        public void addAdmin(Admin admin);

        public void deleteAdmin(Admin admin);

        public List<Admin> searchAdmins(String search);

        public Boolean existsAdmin(Admin admin);

        public Admin getAdmin(Admin admin);

        public void addFavUsers(Admin admin, User user);

}
```

- **public List<Admin> getAdmins():** Returns the whole list of Admins with a simple *SELECT* query.
- **public void addAdmin(Admin admin):** Inserts the given Admin to the Admin Table with a simple *INSERT INTO* query.
- **public void deleteAdmin(Admin admin):** Removes the given Admin from the Admin table with a simple *DELETE* query.
- **public List<Admin> searchAdmins(String search):** Searches admins, whose name starts with the given string, with a simple *SELECT* query.
- **public Boolean existsAdmin(Admin admin):** Returns '1' if there is an admin with the given admins name and password, otherwise returns '0'.
- **public Admin getAdmin(Admin admin):** Returns the information of Admin, which is sent to this function as parameter.
- **public void addFavUsers(Admin admin, User user):** Inserts a new row to FAVUSERS Table, with the username of the given admin and the username of the given user. This operation is done with a simple *INSERT INTO* query.